Algoritmos Gulosos (Greedy): Escalonamento de Tarefas.

IMPLEMENTAÇÃO E TEMPO DE EXECUÇÃO:

- Podemos fazer o algoritmo rodar em $O(n \log n)$.
- Começamos ordenando as n requisições pelo tempo de finalização e marcando cada uma nesta ordem, ou seja, assumimos que $f(i) \leq f(k)$, quando $i \leq k$. Essa ordenação leva o tempo $O(n \log n)$.
- Com um tempo adicional de $O(n)$, construímos um arranjo $S[1 \ldots n]$ com a propriedade que $S[i]$ contém o valor de $s(i)$.
- Depois selecionamos as requisições processando os intervalos na ordem crescente de $f(i)$.
- Sempre selecionamos o primeiro intervalo, e iteramos pelos intervalos até obter o primeiro intervalo compatível, onde $s(j) \geq f(1)$. E também selecionamos este intervalo.
- De forma geral, se o intervalo mais recente selecionado termina no tempo f, continuamos a iterar pelo intervalo até obter o primeiro j para o qual $s(j) \geq f$.
- Desta forma implementamos o algoritmo greedy com apenas uma passagem por todos os intervalos em tempo $O(n)$.
- Ou seja, a complexidade do algoritmo é $O(n \log n + n) \Rightarrow O(n \log n)$.

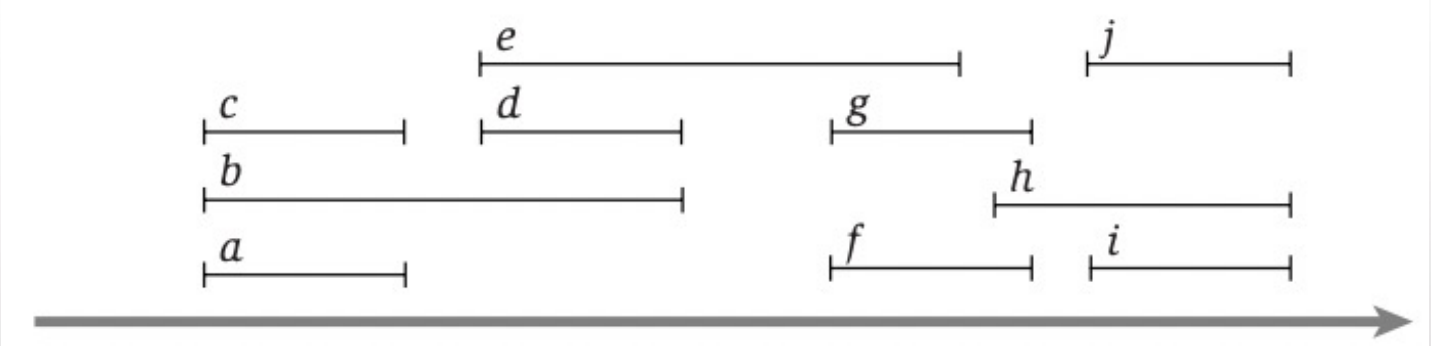
```
EARLIEST-FINISH-TIME-FIRST (n, s1, s2, …, sn , f1, f2, …, fn)

SORT jobs by finish times and renumber so that f1 ≤ f2 ≤ … ≤ fn.
S ← ∅.   ← set of jobs selected
FOR j = 1 TO n
    IF (job j is compatible with S)
        S ← S ∪ { j }.
RETURN S.
```

Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

- No problema de Escalonamento de Tarefas, existe apenas um recurso e muitas requisições na forma de intervalos, de forma que precisamos definir quais requisições aceitar e quais rejeitar.
- Um problema similar emerge se temos vários recursos idênticos e desejamos atender a todas as requisições alocando o menor número de recursos possível.
- Como a atividade é distribuir as requisições por múltiplos recursos, este problema é chamado de Escalonamento de Todas as Tarefas (Interval Partitioning Problem).
- Suponha que cada requisição corresponde a uma aula que precisa ser agenda numa sala por um período específico de tempo.
- Desejamos satisfazer todas as requisições utilizando o menor número de salas possível.
- As salas de aula são os nossos recursos e a restrição básica é que se duas ou mais aulas ocorrerem no mesmo período, elas devem ser alocadas em salas diferentes.
- Outra forma de exemplificar o mesmo problema seria uma fila de “jobs” com um número limitado de máquinas para executá-los.

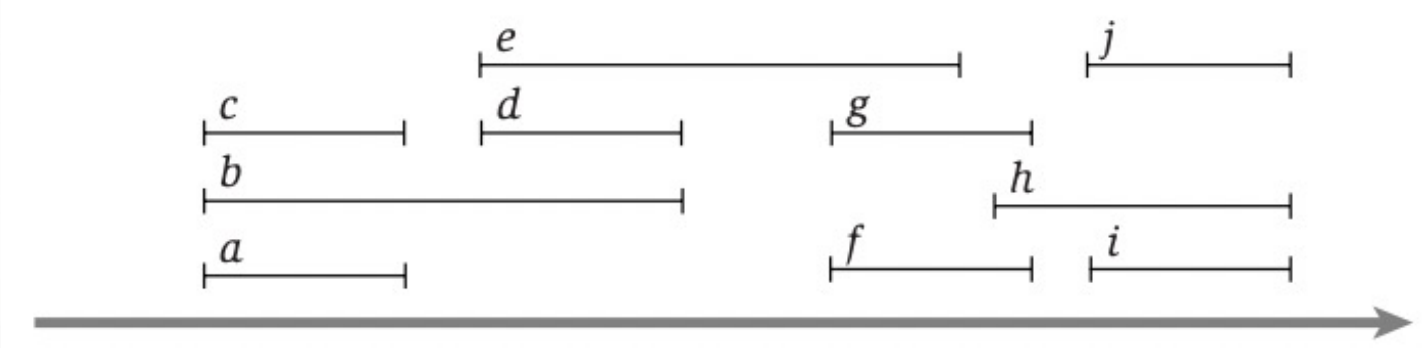
Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

- A figura acima é uma representação do problema:



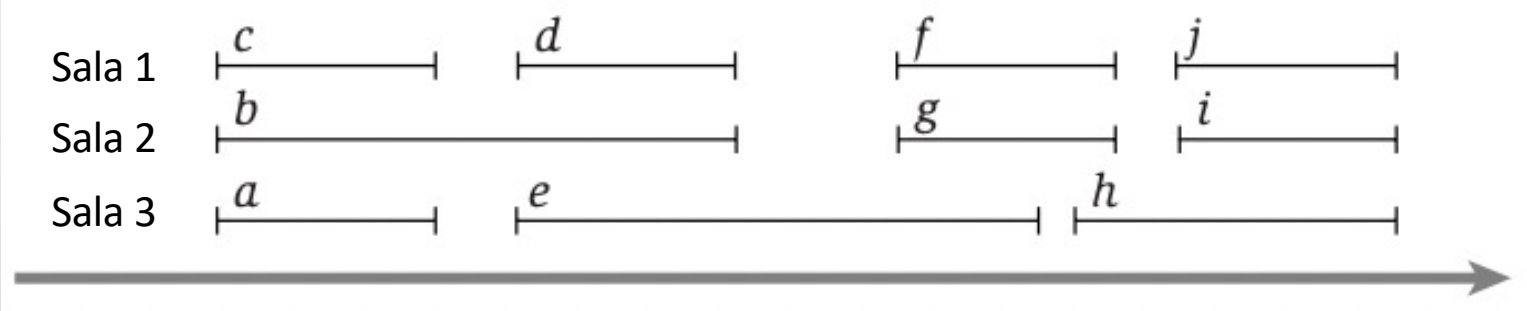
- Quantos recursos seriam necessários para atender todas as requisições?

Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

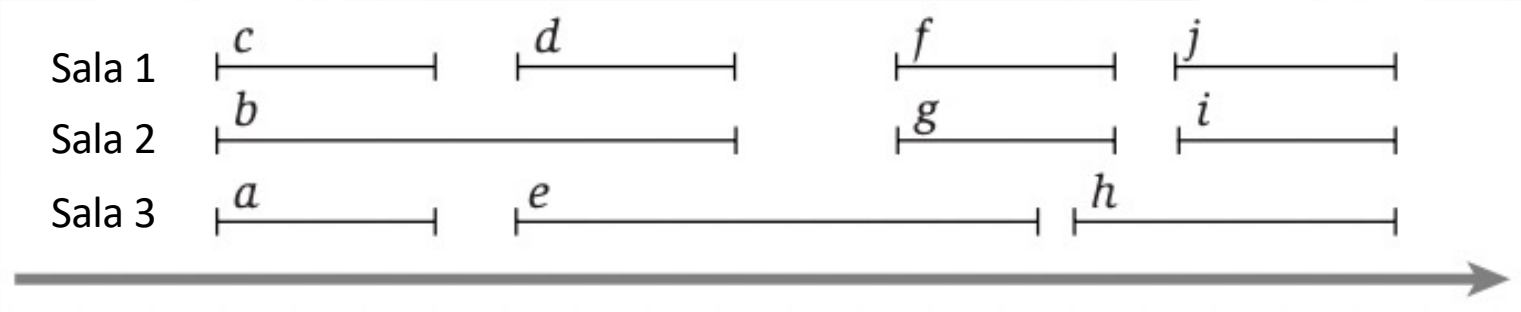


- Quantos recursos seriam necessários para atender todas as requisições?

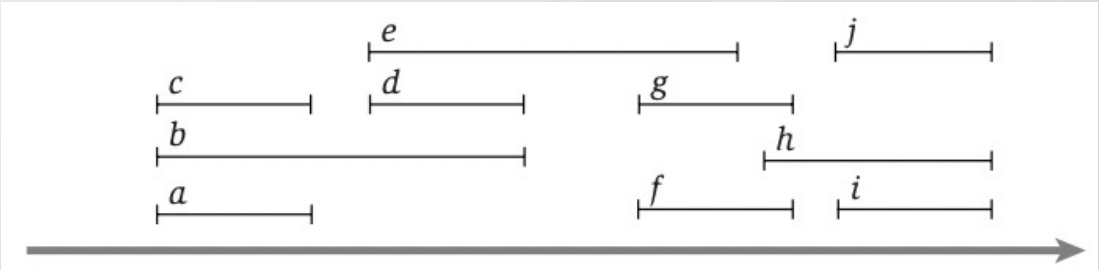
- Todas as requisições podem ser atendidas por três recursos conforme a figura abaixo. Onde as requisições são organizadas em 3 linhas, cada uma contendo intervalos não sobrepostos.



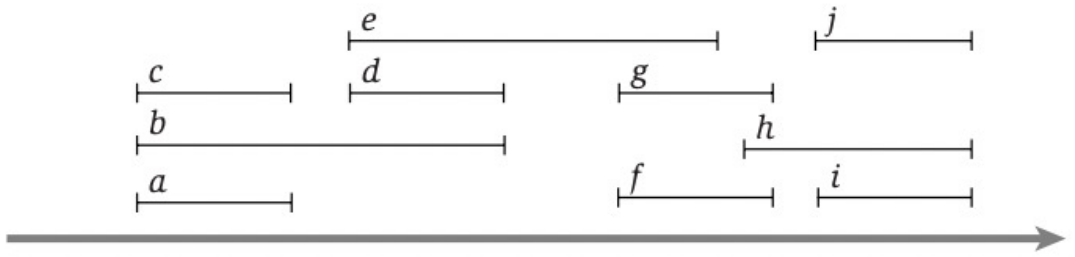
Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.



- De forma geral, podemos imaginar a solução usando k recursos como uma reorganização das requisições em k linhas de intervalos não sobrepostos: a primeira linha contém todas as solicitações atendidas pela Sala 1, a segunda linha todas as requisições alocadas na Sala 2, ...
- Mas existe alguma possibilidade de, por exemplo, apenas 2 recursos atenderem a todas as requisições? Podemos ser mais eficientes?



Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.



- Claramente a resposta é não, pois precisamos de no mínimo 3 recursos para atender as requisições a, b e c de forma simultânea.
- Este argumento pode ser generalizado para qualquer problema de Escalonamento de Todas as Tarefas.
- Vamos definir a profundidade (depth) de um conjunto de intervalos como o número máximo de requisições que passam sobre o mesmo ponto na linha do tempo.

A.4 Em qualquer instância do problema de Escalonamento de Todas as Tarefas o número de recursos necessários é no mínimo igual a profundidade do conjunto de intervalos.

- Suponha que um conjunto de intervalos possua profundidade d, e assuma que $I_1, \ldots, I_d$ passam sobre um ponto comum na linha de tempo. Então cada um desses intervalos deve ser alocado num recurso diferente, de forma que o problema precisara de no mínimo d recursos.

Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

- Podemos desenhar um algoritmo eficiente que aloque todas as solicitações com o menor número de recursos?
- Sempre existe uma alocação usando o número de recursos igual a profundidade d?
- Se a reposta para a segunda for positiva, isso significa que os obstáculos para escalonar todas as tarefas são puramente locais, ou seja, um grupo de requisições empilhados no mesmo período precisam ser distribuídos.
- Porém, não está claro que não poderiam existir outros obstáculos que elevem o número de recursos necessários.

Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

O ALGORITMO:

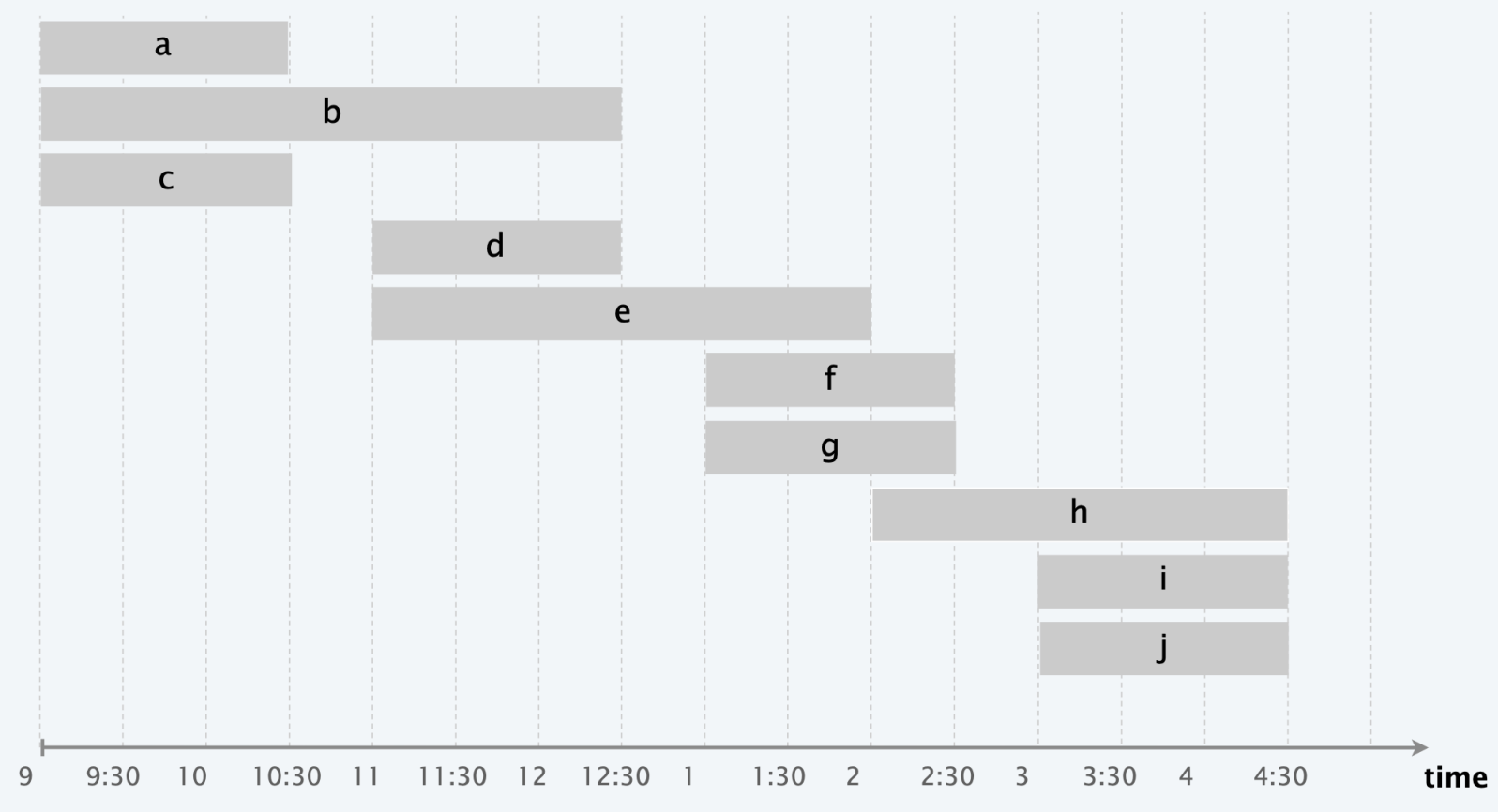
- Vamos construir um algoritmo que aceita todas as requisições utilizando o numero de recursos igual a d.
- Considerando A.4, essa abordagem já implica que o algoritmo é ótimo, ou seja, nenhuma solução pode utilizar menos recursos que d.
- Associe um rótulo para cada requisição onde os rótulos são oriundos do conjunto $\{1,2, \ldots, d\}$ e o processo de associação garante que requisições sobrepostas recebem rótulos com diferentes números.
- Com isso chegamos a solução desejada, pois podemos interpretar cada número como o nome de um recurso diferente, e o rótulo de cada solicitação como o nome do recurso para o qual ela foi alocada.
- O algoritmo é uma estratégia greedy simples de uma passada que ordena os intervalos pelo seu tempo de início $s(i)$.
- Analisamos as requisições nesta ordem e rotulamos cada requisição com um rótulo diferente das requisições anteriores que ocorrem ao mesmo tempo.

```
Sort the intervals by their start times, breaking ties arbitrarily
Let I_1, I_2, ..., I_n denote the intervals in this order
For j = 1, 2, 3, ..., n
  For each interval I_i that precedes I_j in sorted order and overlaps it
     Exclude the label of I_i from consideration for I_j
  Endfor
  If there is any label from {1, 2, ..., d} that has not been excluded then
    Assign a nonexcluded label to I_j
  Else
    Leave I_j unlabeled
  Endif
Endfor
```

Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

EXEMPLO

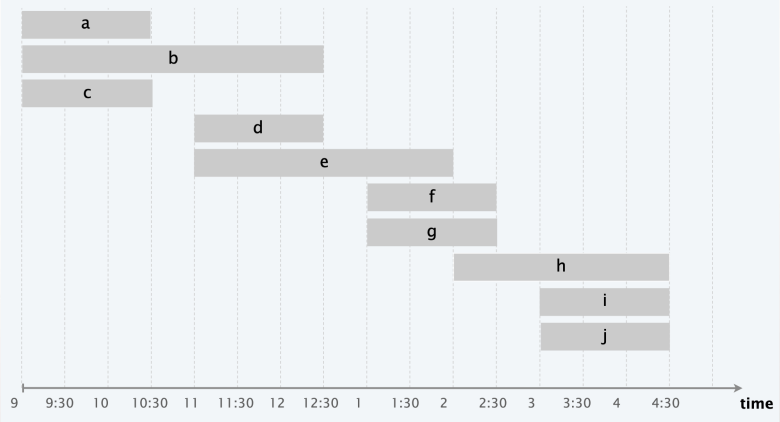
- Quantos recursos são necessários para alocar todas as atividades abaixo?
- Liste quais atividades são alocadas em cada recurso.

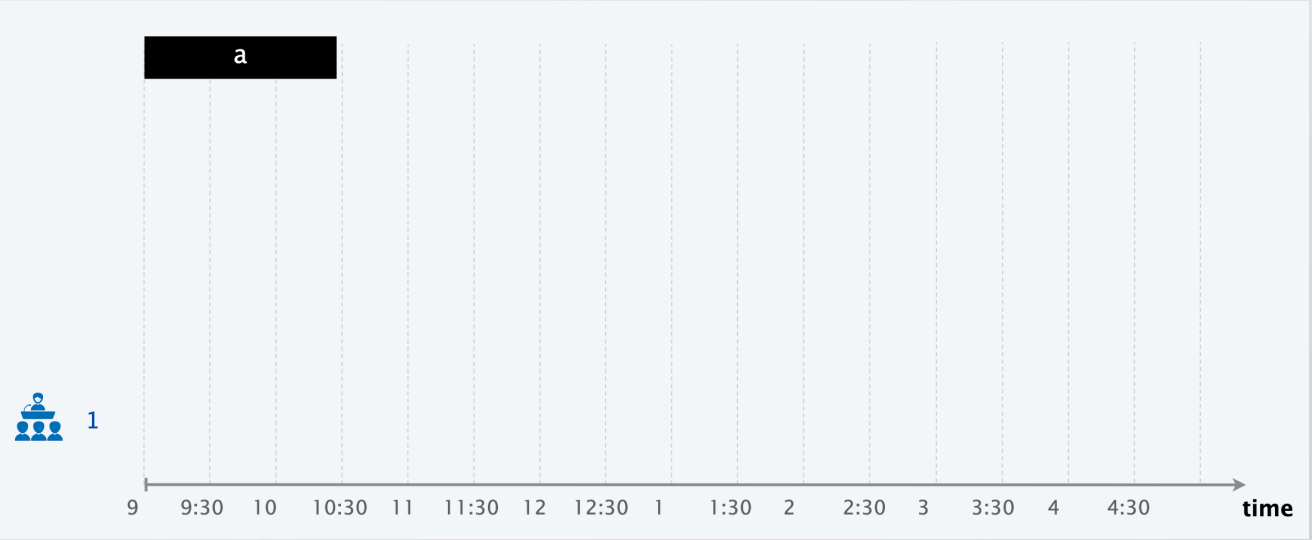


Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

EXEMPLO

- Quantos recursos são necessários para alocar todas as atividades abaixo?
- Liste quais atividades são alocadas em cada recurso.



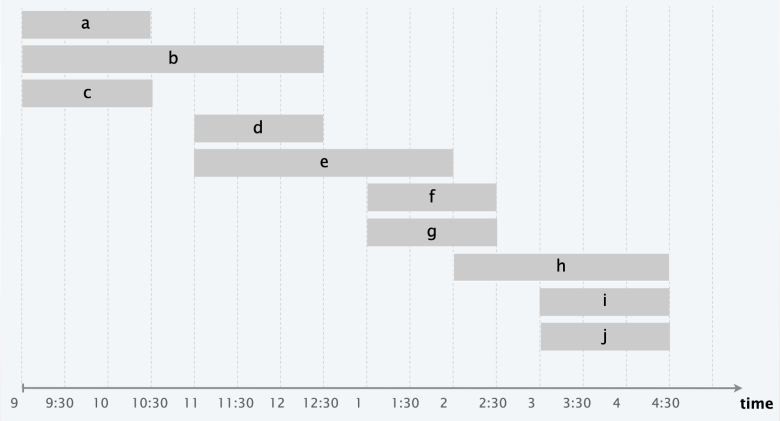


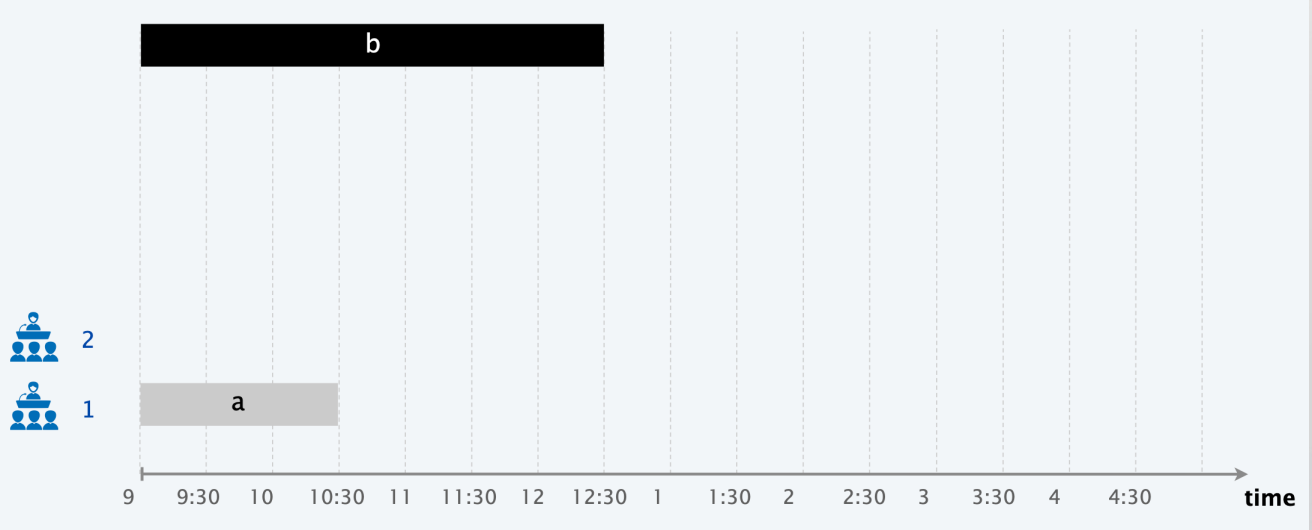
PUCRS ESCOLA POLITÉCNICA

Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

EXEMPLO

- Quantos recursos são necessários para alocar todas as atividades abaixo?
- Liste quais atividades são alocadas em cada recurso.

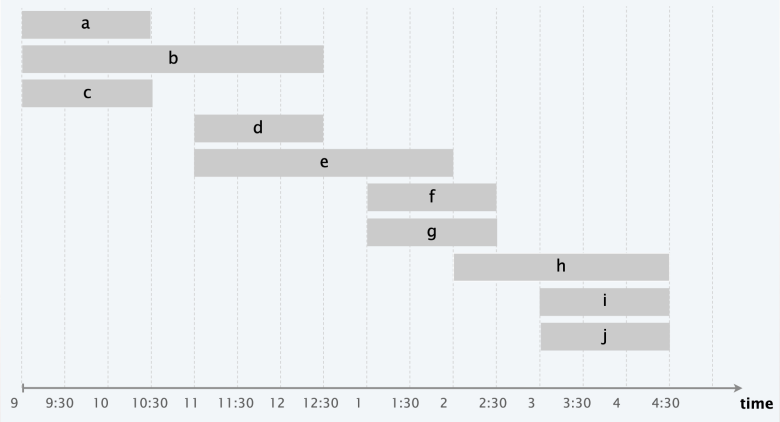


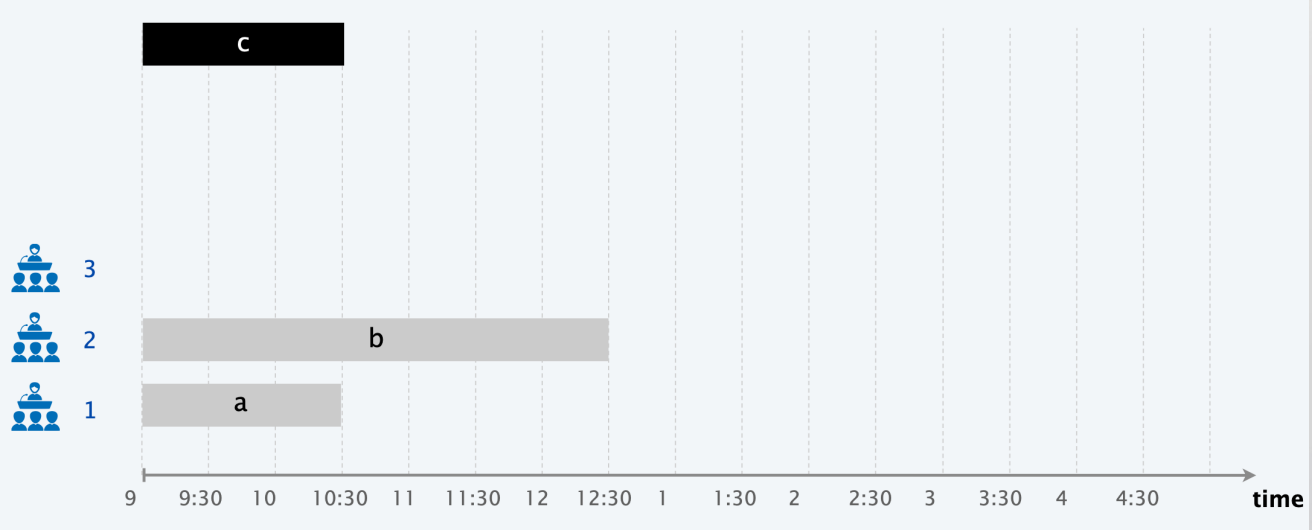


Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

EXEMPLO

- Quantos recursos são necessários para alocar todas as atividades abaixo?
- Liste quais atividades são alocadas em cada recurso.



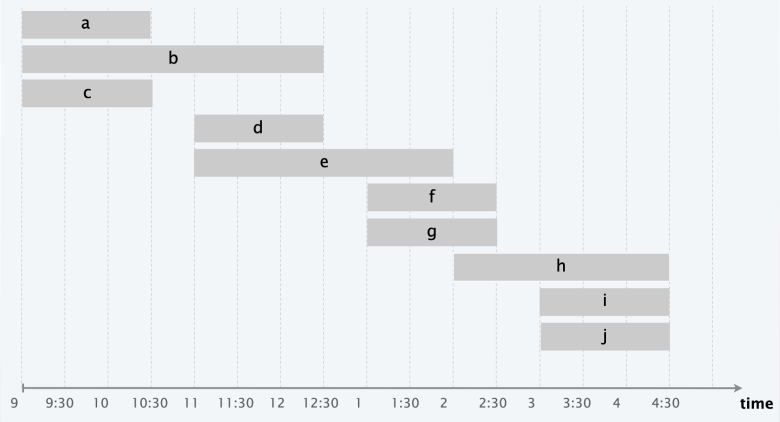


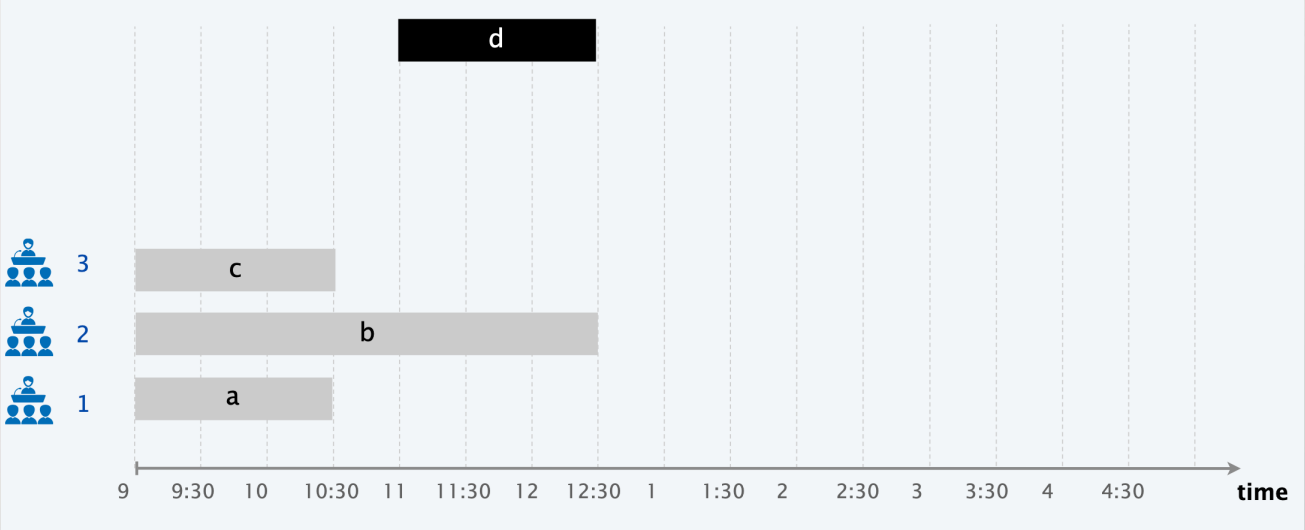
PUCRS ESCOLA POLITÉCNICA

Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

EXEMPLO

- Quantos recursos são necessários para alocar todas as atividades abaixo?
- Liste quais atividades são alocadas em cada recurso.

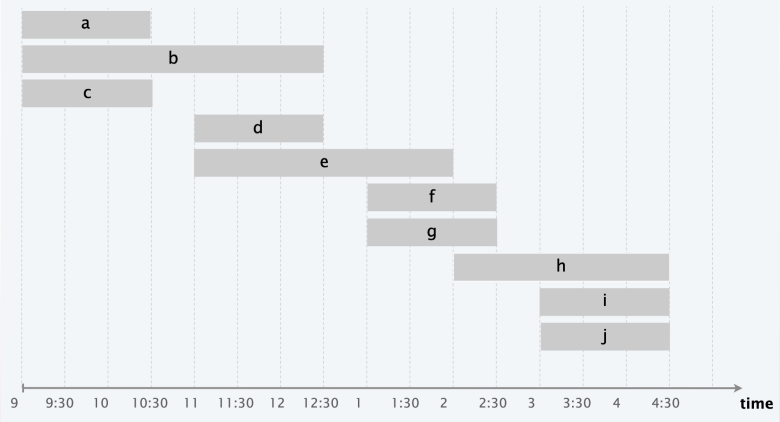


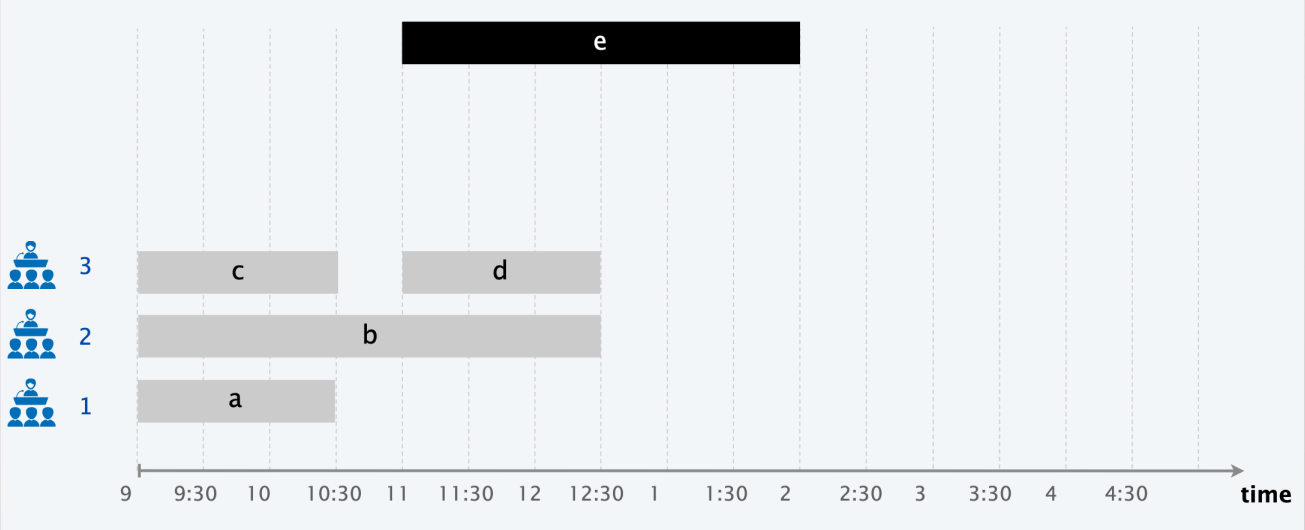


Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

EXEMPLO

- Quantos recursos são necessários para alocar todas as atividades abaixo?
- Liste quais atividades são alocadas em cada recurso.



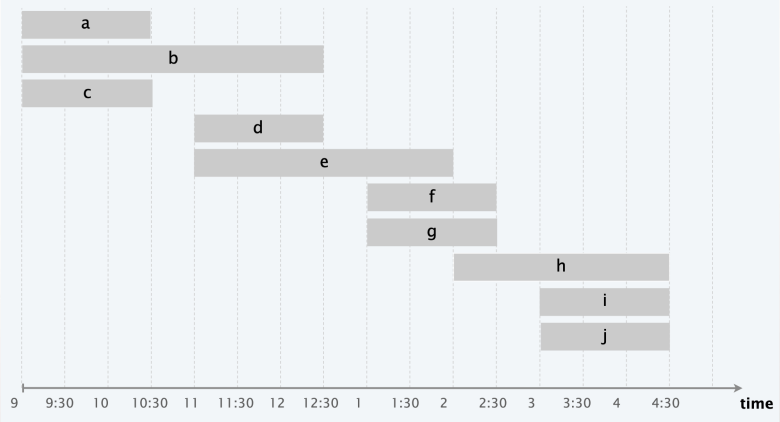


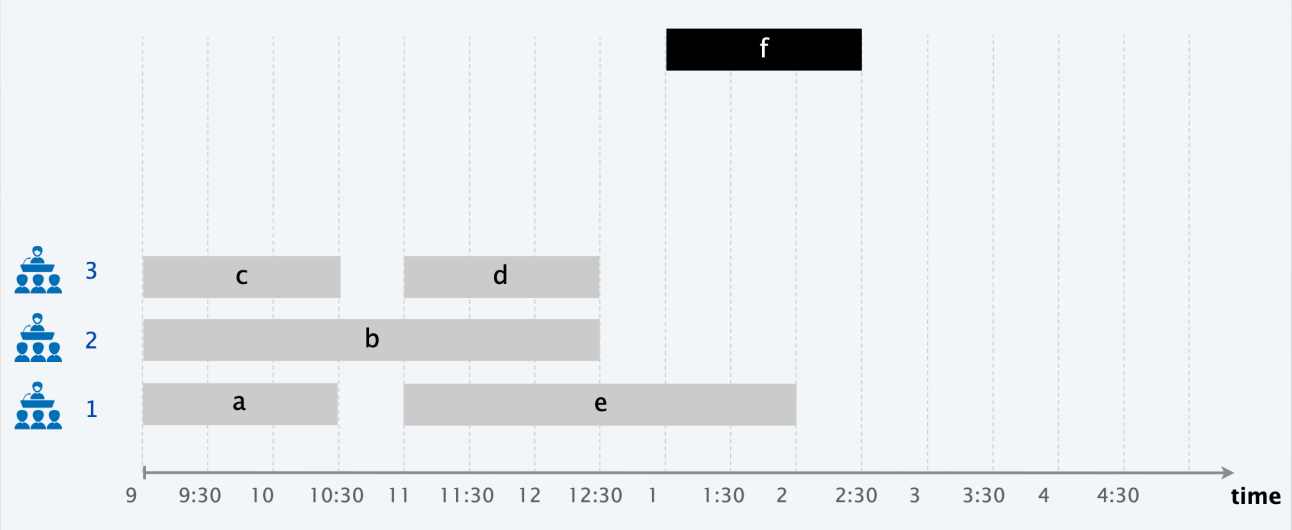
PUCRS ESCOLA POLITÉCNICA

Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

EXEMPLO

- Quantos recursos são necessários para alocar todas as atividades abaixo?
- Liste quais atividades são alocadas em cada recurso.



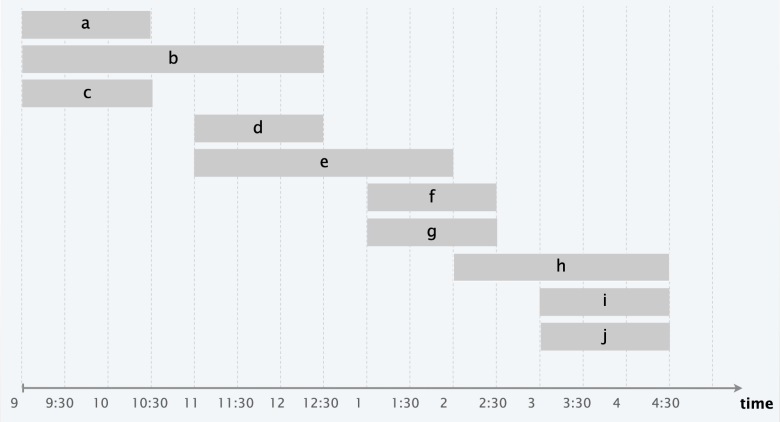


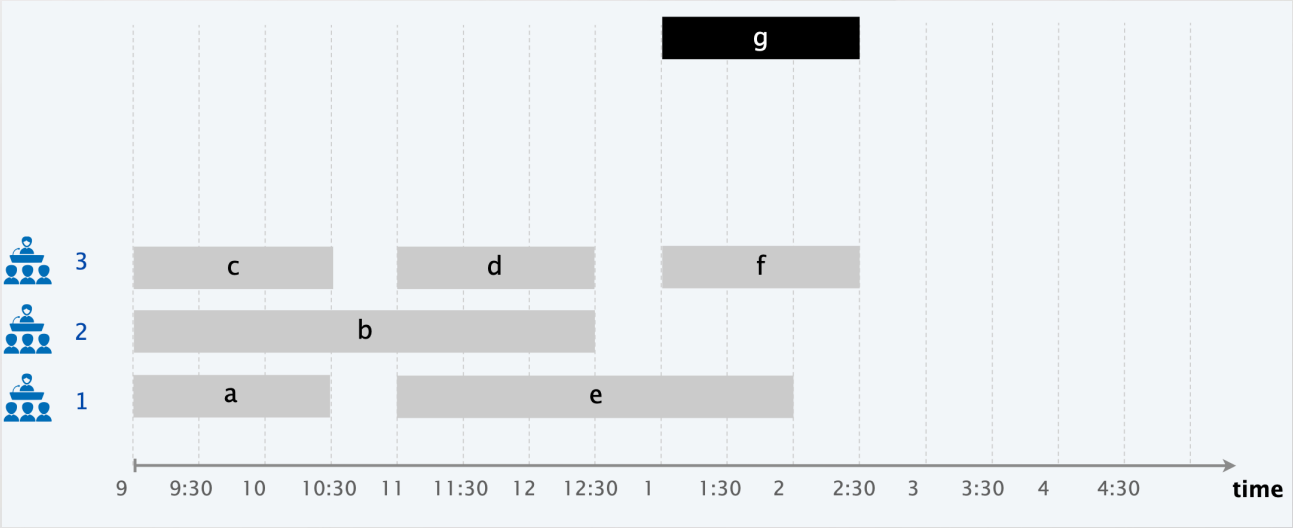
PUCRS ESCOLA POLITÉCNICA

Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

EXEMPLO

- Quantos recursos são necessários para alocar todas as atividades abaixo?
- Liste quais atividades são alocadas em cada recurso.



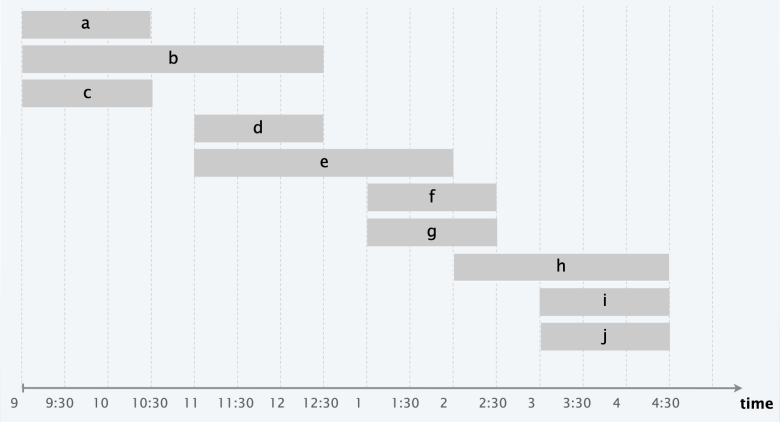


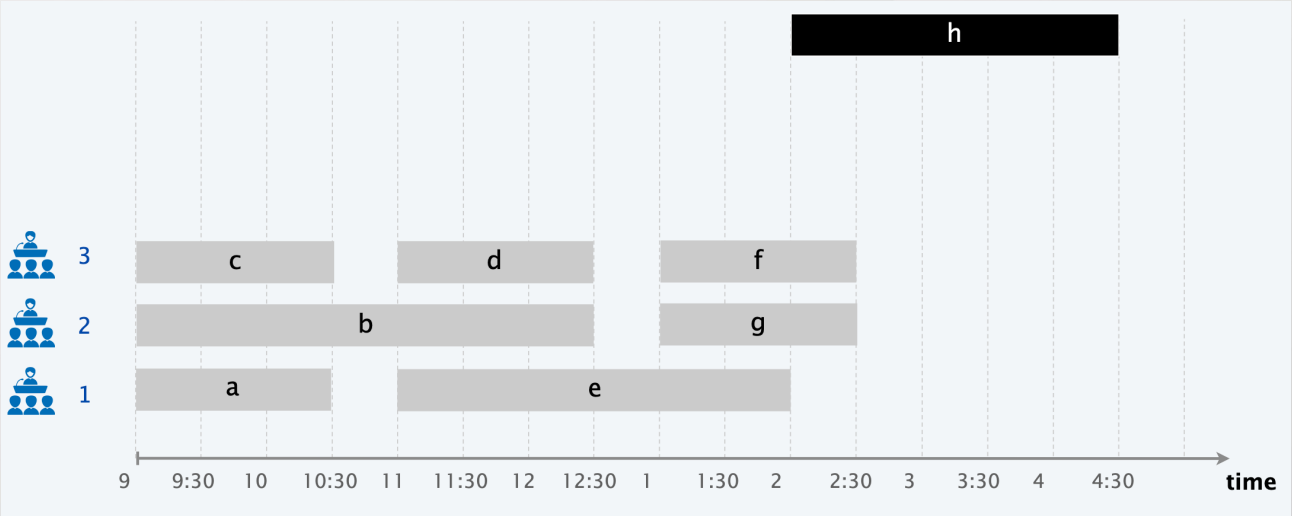
PUCRS ESCOLA POLITÉCNICA

Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

EXEMPLO

- Quantos recursos são necessários para alocar todas as atividades abaixo?
- Liste quais atividades são alocadas em cada recurso.

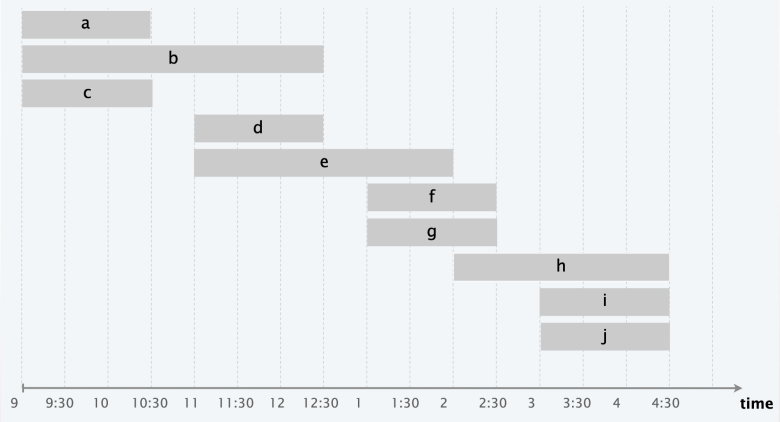


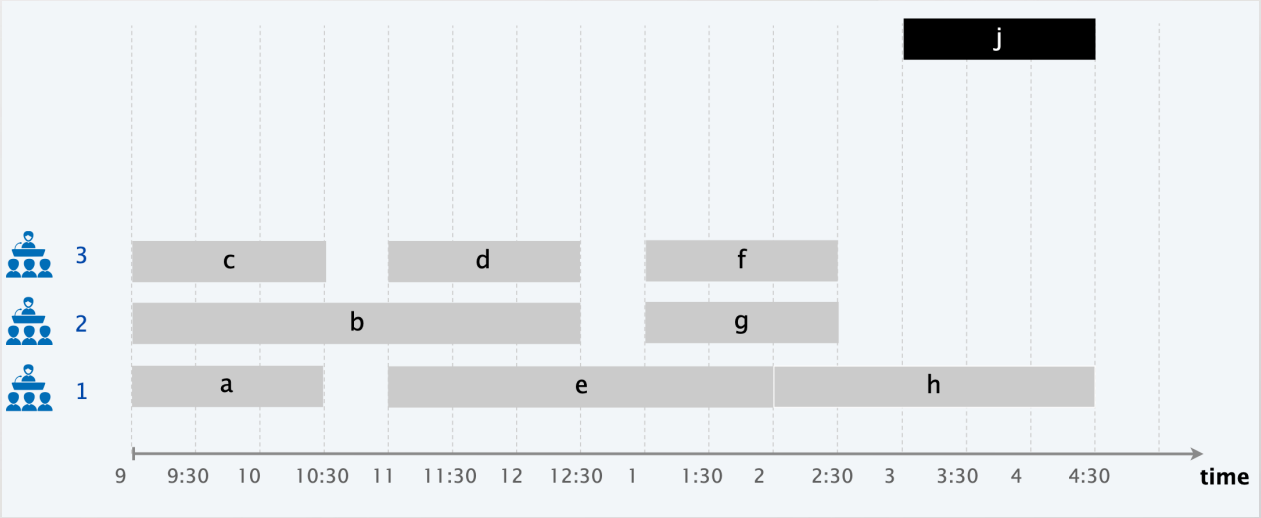


Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

EXEMPLO

- Quantos recursos são necessários para alocar todas as atividades abaixo?
- Liste quais atividades são alocadas em cada recurso.



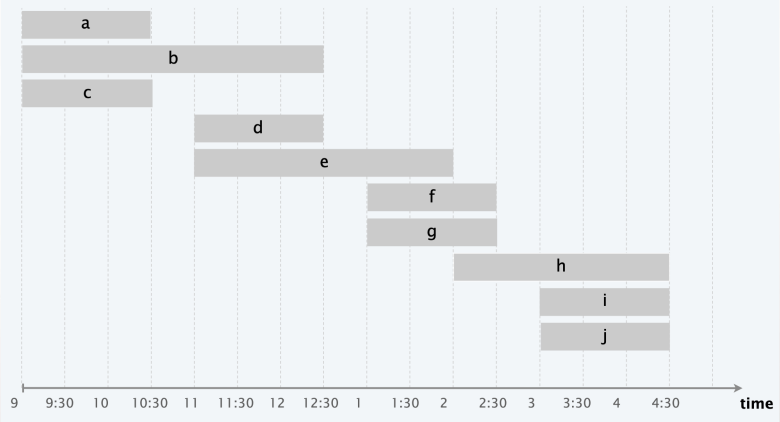


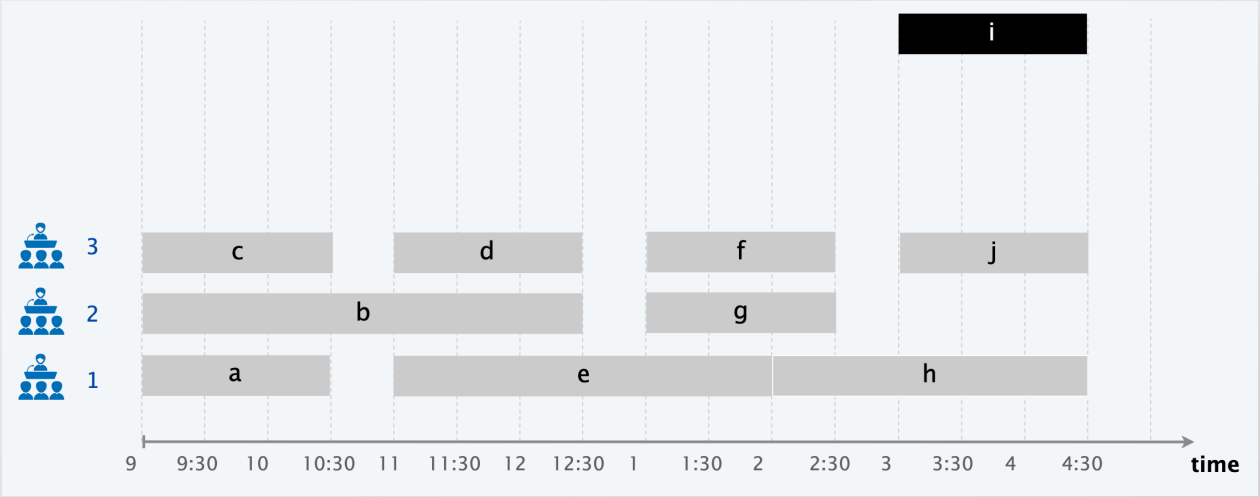
PUCRS ESCOLA POLITÉCNICA

Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

EXEMPLO

- Quantos recursos são necessários para alocar todas as atividades abaixo?
- Liste quais atividades são alocadas em cada recurso.

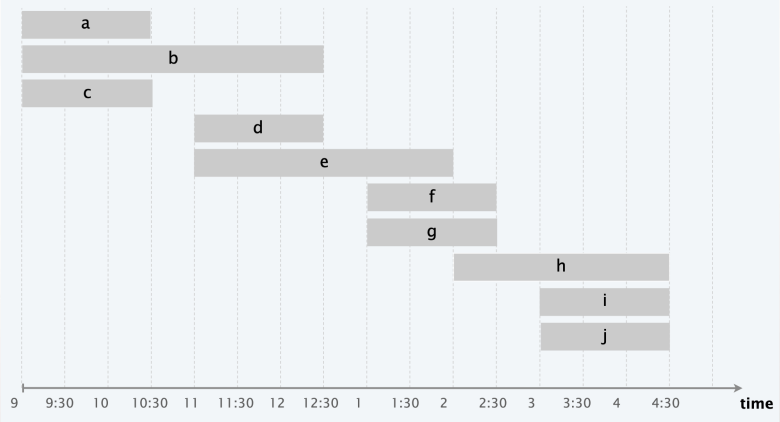


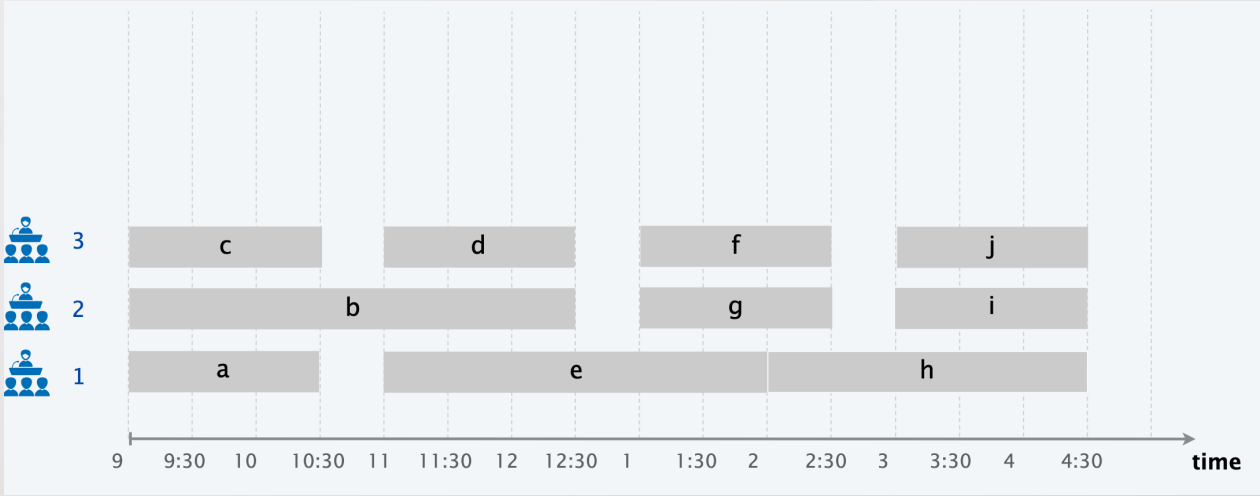


Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

EXEMPLO

- Quantos recursos são necessários para alocar todas as atividades abaixo?
- Liste quais atividades são alocadas em cada recurso.





Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

ANALISANDO O ALGORITMO:

A.5 Para o algoritmo especificado, cada intervalo receberá um rótulo e, intervalos que se sobreponham não receberão o mesmo rótulo.

- Primeiramente, vamos argumentar que nenhuma requisição ficará sem rótulo.
- Considere uma das requisições I_j e suponha que existem t requisições antes no conjunto ordenado de requisições que sobrepõem I_j .
- Essas t requisições, juntamente com I_j, formam um conjunto de $t + 1$ intervalos que passam sobre um mesmo ponto comum na linha temporal (ou seja, o ponto de início $s(j)$ de I_j), e portanto, $t + 1 \leq d$.
- Logo, $t \leq d - 1$, ou seja, pelo menos um dos d rótulos não é excluído pelo conjunto de t requisições, e portanto, existe um rótulo d que pode ser alocado em I_j.
- Depois, afirmamos que duas requisições sobrepostas não podem receber o mesmo rótulo.
- Considere duas requisições I e I' que se sobrepõem, e suponha que I precede I' quando ordenados pelo tempo inicial.
- Então quando I' é considerado pelo algoritmo, I está no conjunto de requisições cujos rótulos não são considerados, consequentemente, o algoritmo não irá atribuir a I' o rótulo de I.

```
Sort the intervals by their start times, breaking ties arbitrarily
Let I_1, I_2, ..., I_n denote the intervals in this order
For j = 1, 2, 3, ..., n
  For each interval I_i that precedes I_j in sorted order and overlaps it
    Exclude the label of I_i from consideration for I_j
  Endfor
  If there is any label from {1, 2, ..., d} that has not been excluded then
    Assign a nonexcluded label to I_j
  Else
    Leave I_j unlabeled
  Endif
Endfor
```

Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

- O algoritmo e sua análise são muito simples.
- Essencialmente, possuímos d rótulos a disposição, e passamos por todas as requisições ordenadas por ordem de início da esquerda para a direita. Rotulamos cada requisição que encontramos e sempre teremos rótulos para todas as requisições.
- Como nosso algoritmo sempre usa d rótulos, podemos concluir usando A.4 que:

A.6 O algoritmo greedy aloca todas as requisições para um recurso, usando um número de recursos igual a profundidade do conjunto de requisições. Este é o número ótimo de recursos utilizados.

- A análise que realizamos nesta aula ilustra uma outra abordagem para provar que um algoritmo é ótimo. Primeiramente, encontramos um limite "estrutural" e simples, afirmando que cada solução deve possuir ao menos um certo valor (neste caso d) e depois demonstramos que o algoritmo sendo considerado sempre atinge este limite.

Algoritmos Gulosos (Greedy): Escalonamento de Todas Tarefas.

IMPLEMENTAÇÃO E TEMPO DE EXECUÇÃO:

- Podemos fazer o algoritmo rodar em $O(n \log n)$.

EARLIEST-START-TIME-FIRST $(n, s_1, s_2, \ldots, s_n, f_1, f_2, \ldots, f_n)$

SORT lectures by start times and renumber so that $s_1 \leq s_2 \leq \ldots \leq s_n$.

$d \leftarrow 0$. ← number of allocated classrooms

FOR $j = 1$ TO n

  IF (lecture j is compatible with some classroom)

    Schedule lecture j in any such classroom k.

  ELSE

    Allocate a new classroom $d + 1$.

    Schedule lecture j in classroom $d + 1$.

    $d \leftarrow d + 1$.

RETURN schedule.